条件语句

格式一

if 判断条件:
    执行代码块

```
age = float(input('请问你今年多少岁? '))
if age >= 18:
    print('你已经成年了!')
```

格式二

if 判断条件:
    执行代码块1
else:
    执行代码块2

```
age = float(input('请问你今年多少岁? '))
if age >= 18:
    print('你已经成年了!')
else:
    print('你还未成年!')
```

格式三

if 判断条件1:
    执行代码块1
elif 判断条件2:
    执行代码块2
elif 判断条件3:
    执行代码块3
...
else:
    执行代码块n

```
score = float(input('你这次考试考了多少分? '))
if score >= 90:
    print('厉害!')
elif score >= 80:
    print('优秀!')
elif score >= 70:
    print('良好!')
elif score >= 60:
    print('及格!')
else:
    print('不及格!')
```

三元表达式

- 三元表达式用来实现一些简单的条件语句，会比结构化的代码块更灵活

```
age = float(input('请问你今年多少岁? '))
print('你已经成年了!') if age >= 18 else print('你还未成年!')


score = float(input('你这次考试考了多少分? '))
print('厉害!') if score >= 90 else \
print('优秀!') if score >= 80 else \
print('良好!') if score >= 70 else \
print('及格!') if score >= 60 else \
```

```
print('不及格!')



score = float(input('你这次考试考了多少分? '))
res = '厉害!' if score >= 90 else \
      '优秀!' if score >= 80 else \
      '良好!' if score >= 70 else \
      '及格!' if score >= 60 else \
      '不及格!'
print(res)
```

条件语句嵌套

```
age = float(input('请问你今年多少岁? '))
if age >= 18:
    ans = input('可以出示您的身份证吗(Y/N): ')
    if ans == 'Y':
        print('身份核对正确, 请尽情冲浪吧!')
    else:
        print('成年人需要凭身份证上网!')
else:
    print('未成年人禁止出入网吧!')
```

条件语句特点

- 每个条件语句中，最多只会满足一次条件

```
num = 5

if num > 0:
    print('a')
elif num > 1:
    print('b')
elif num > 2:
    print('c')
if num > 3:
    print('d')
if num > 4:
    print('f')
if num > 5:
    print('g')
else:
    print('h')
```

- 当判断条件是一个值时，该值bool判定的结果决定条件是否成立

```
if 1:
    print('hello world')

if None:
    print('hello world')
```

简单的猜拳游戏

```
d = {'石头': 0, '剪刀': 1, '布': 2}

computer = set(d).pop()
player = input('请出拳(石头、剪刀、布): ')

# 出拳展示
print(f'电脑出拳: {computer}\n玩家出拳: {player}')

# 胜负判定
c, p = d[computer], d[player]
if p-c in (-1, 2):
    print('玩家胜!')
elif c == p:
    print('平局!')
else:
    print('电脑胜!')
```

循环语句

while 循环

while 判断条件:
    循环体

```
# 输出1-100之间的整数
num = 1
while num <= 100:
    print(num)
    num += 1
```

while True

- 无限循环（俗称：死循环），通常配合 break 使用

```
# 输出1-100之间的整数
num = 1
while True:
    print(num)
    num += 1
    if num > 100:
        break
```

while 循环嵌套

```
# 实现九九乘法表
right = 1
while right <= 9:
    left = 1
    while left <= right:
        print(f'{left}x{right}={left*right}',
end='\t')
        left += 1
    print()
    right += 1
```

for 循环

for 变量 in 可迭代对象:
    循环体

```
"""
第1次循环: i = lst[0], 执行循环体 print(i)
第2次循环: i = lst[1], 执行循环体 print(i)
第3次循环: i = lst[2], 执行循环体 print(i)
第4次循环: i = lst[3], 执行循环体 print(i)
当取不到lst中元素时，for循环自动停止。
"""
lst = ['d', 'c', 'k', 'a']
# 获取元素
for i in lst:
    print(i)
```

```python
# 获取索引
for i in range(len(lst)):
    print(i)

# 获取索引和元素
for index, item in enumerate(lst):
    print(index, item)
```

for 循环嵌套

```python
# 实现九九乘法表
for right in range(1, 10):
    for left in range(1, right+1):
        print(f'{left}x{right}={left*right}',
end='\t')
    print()
```

range([start], stop[, step])

- 按照 step 生成从 start 到 stop 的整数序列（不可变）
- start：起始值，闭区间，默认为 0
- stop：结束值，开区间
- step：步长，默认为 1

```
print(list(range(4)))  # [0, 1, 2, 3]
print(list(range(1, 5)))  # [1, 2, 3, 4]
print(list(range(1, 8, 2)))  # [1, 3, 5, 7]
print(list(range(8, 1, -2)))  # [8, 6, 4, 2]

rg = range(1, 8, 2)
print(len(rg))  # 4
print(rg[2])  # 5
print(rg[::2])  # range(1, 9, 4)
```

enumerate(iterable, start=0)

- 返回一个迭代器对象。迭代它会得到一个个的元组，每个元组是由索引和对应元素构成的。start决定索引的起始值。

```
lst = ['d', 'c', 'k', 'a']
print(list(enumerate(lst)))
print(tuple(enumerate(lst)))

for i in enumerate(lst):
    print(i)
```

循环控制语句

break

- 终止所在的循环

```
for _ in range(3):
    for _ in range(4):
        print('hello')
        break


for _ in range(3):
    for _ in range(4):
        print('hello')
    break
```

continue

- 跳过当前这次循环，继续到下一次循环

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]

for number in numbers:
    if number % 2 == 0:
        continue  # 如果数字是偶数，跳过当前循环
    print(number)
```

推导式

列表推导式

- 格式：[exp for子句]
- 格式：[exp for子句 更多的for子句或者if子句]

```
lst = [x ** 2 for x in range(4)]
print(lst)

# 类比
lst = []
for x in range(4):
    lst.append(x ** 2)
print(lst)



lst = [x + y for x in range(5) if x % 2 for y in
(1, 2, 3)]
print(lst)

# 类比
lst = []
for x in range(5):
    if x % 2:
        for y in (1, 2, 3):
            lst.append(x + y)
print(lst)
```

字典推导式

- 格式：{k: v for子句}
- 格式：{k: v for子句 更多的for子句或者if子句}

```
d = {x: x**2 for x in range(4)}
print(d)

# 类比
d = {}
for x in range(4):
    d[x] = x ** 2
print(d)



d = {x: v for x in range(4) for v in range(9) if
v % 2}
print(d)

# 类比
d = {}
for x in range(4):
    for v in range(9):
        if v % 2:
            d[x] = v
print(d)
```

集合推导式

- 格式：{exp for子句}
- 格式：{exp for子句 更多的for子句或者if子句}

```
s = {x ** 2 for x in range(4)}
print(s)

# 类比
s = set()
for x in range(4):
    s.add(x ** 2)
print(s)



s = {x + y for x in range(5) if x % 2 for y in
(1, 2, 3)}
print(s)

# 类比
s = set()
for x in range(5):
    if x % 2:
        for y in (1, 2, 3):
            s.add(x + y)
print(s)
```

pass 语句

pass 是一个关键字，表示一个空语句，当它被执行时，不做任何操作，通常用作占位语句，在语法上需要语句但实际上不需要执行任何操作的情况下使用。

```
score = float(input('你这次考试考了多少分? '))
if score >= 90:
    print('厉害!')
elif score >= 80:
    pass
elif score >= 70:
    print('良好!')
elif score >= 60:
    ...
else:
    print('不及格!')
```